ARPA

Command Systems Cybernetics

LEVEL III

12

A061206

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

LEVEL II

(12)

Contract Period Covered by Report: 1 November 1977 - 28 February 1978

(9) Quarterly Research and Development Technical Report.

(6) Spatial Data Management System.

Computer Corporation of America

1 Nov 77 - 28 Feb 78

The views and conclusions in this document are those of the authors and should not be interpreted as necessarily representing the official policies, express or implied, of the Advanced Research Projects Agency, or the United States Government.

Report Authors: (10) Christopher F. Herot, James B. Rothnie, Jr., Jerry Farrell

(11) 1978

(12) 51p.

Research Division
Computer Corporation of America, Cambridge, MA

617-491-3670

Sponsor: Defense Advanced Research Projects Agency, Office of Cybernetics Technology

ARPA Order Number: 3487

ARPA Contract Number: (15) MDA903-78-C-0122, ARPA Order-3487

Contract Period: 15 February 1978 - 30 November 1979

387285

VB

Table of Contents

1. INTRODUCTION

This report describes the first four months of effort in the design and implementation of a prototype spatial data management system (SDMS). Spatial Data Management is a technique for organizing and retrieving textual, symbolic, and pictorial information by positioning it in an Information Space maintained through the use of interactive computer graphics. The system currently under construction will employ a multiple channel color raster scan display to allow a user to organize his information within nested "planes" of data over which he can maneuver a window by the use of joy sticks.

By allowing a datum to be stored in proximity to related pieces of information, the system frees the user of the need to know the exact name or location of the information he seeks. Instead, he can locate it by "browsing" until he finds something he can identify visually. Such a technique is envisaged to have widespread application in any field where efficient and quick access to large and complex databases is required, such as command, control and communications, intelligence analysis, and the management of any complex activity.

The first quarter of this project, reported briefly herein, has involved the specification of the system capabilities, selection of display hardware, and exploration of several important implementation issues.

1.1 Background

Initial work in the spatial management of data was done at the Massachusetts Institute of Technology under ARPA/CTO sponsorship. [Negroponte and Fields] describe the general features of the MIT system, which was the first existing SDMS. The system was implemented on specially designed and constructed hardware which provided several unique features key to the successful implementation of SDMS, including the ability, in real time, to zoom (magnify) and scroll (move the picture in the plane of the display) so as to give the appearance of flying over a flat world of data.

A user of the MIT system is presented with nested two-dimensional planes of data which he can view on a large (6 ft x 9 ft) color display screen. By exerting pressure on a joystick mounted in the arm of a specially modified chair, he can cause the images on the current level to translate about the screen. These images are

typically hand-constructed icons which indicate the presence of various pieces of data. When he has an icon centered on the screen, the user can view the associated data by pressing on a second joy stick which zooms in to the display. As the magnification of the image increases, the computer replaces the view of the image plane with a new one. This may be the desired information itself, in the form of a document or photograph, or it may be another plane of icons, in which case the process of selection and zooming may be repeated. Most of these planes are larger than the area of the display, such that all of a plane is not visible at once. At the top-most (initial) level, this problem is countered by providing the user with an auxiliary display on which he can view a low resolution image of the entire plane upon which a cursor indicates which portion is currently being shown on the large display.

There is no database management system provided other than that which contains the image planes themselves. Adding new information requires manual intervention in the form of preparing the appropriate icon and formatting the document or photograph. The latter operation is most often accomplished by placing such a document, in paper form, under a vidicon and digitizing it into a raster image.

The database contained in the current system is a set of photographs, maps, and documents describing MIT. This database and the results of MIT's first year of their two year effort is described by [Bolt]. The system which MIT is currently implementing in their second year is described in a forthcoming paper [Donelson].

1.2 The CCA Prototype

In addition to the usual constraints involved in carrying a concept from the laboratory to a prototype environment, the system under construction by CCA is intended to meet several additional criteria:

1. It must run on commercially available hardware to permit ease of replication and simplify maintenance.

2. It must be designed with a certain amount of hardware independence so as to accommodate new capabilities which are appearing rapidly, such as 1000 line television.

3. It needs a good interface to a database management system, to allow the input of large amounts of data without excessive "hand crafting" and to permit a user to select data objects by graphical indications.

The remainder of this report describes the steps which have been taken to design a prototype SDMS which meets the above criteria and that can be applied to a wide range of activities and databases.

Chapter 2 describes how the system will appear to the user. It describes how information is organized in an SDMS, how icons are manipulated and how various subsystems are activated.

Chapter 3 sets forth the requirements imposed on the display hardware by the unusual nature of the SDMS graphic manipulations.

Chapter 4 reports on the results of an analysis of the problem of manipulating the large quantities of information needed to give the user the appearance of motion over a data space.

Chapter 5 specifies how the Spatial Data Management System will interface to a symbolic database management system, such as the INGRES relational database.

2. USER LEVEL VIEW OF SDMS

The user of the prototype SDMS will sit at a display station consisting of three color television monitors, a data tablet, a keyboard, and two joysticks. The largest of the monitors is the primary display of the data space. The two, smaller monitors provide auxiliary views to aid the user in navigating that space.

When first activated, the SDMS presents the user with a view of the "top-level" data plane. The main monitor indicates what information is available on that plane as a set of icons. These icons may be very simple, such as colored rectangles with strings of text, or they may be very elaborate, such as pictures of ships or peoples faces reproduced from photographs.

The joysticks permit the user to move about the data plane. One of them controls horizontal motion - pressing it causes the image on the screen to move horizontally or vertically, as if the user was flying a helicopter at a constant altitude above the plane of data. If the plane being viewed is larger than the display screen, this motion will cause new icons to appear at one margin as old ones move off the opposite one.

To help the user understand where he is in the data plane, one of the smaller monitors provides a low resolution picture of the entire data plane. This display never scrolls. Rather, it remains stationary as a cursor moves over it, indicating the user's position in the data plane.

The second joystick controls the magnification of the display. Pressing it causes the image to "zoom" almost as if the user's "helicopter" began to descend. As this effect is accomplished by changing the readout of the display memory, the picture does not increase in detail, but the icons on the screen do appear bigger. At the same time, the cursor on the auxiliary monitor becomes smaller, indicating that a smaller portion of the data plane is visible on the main screen.

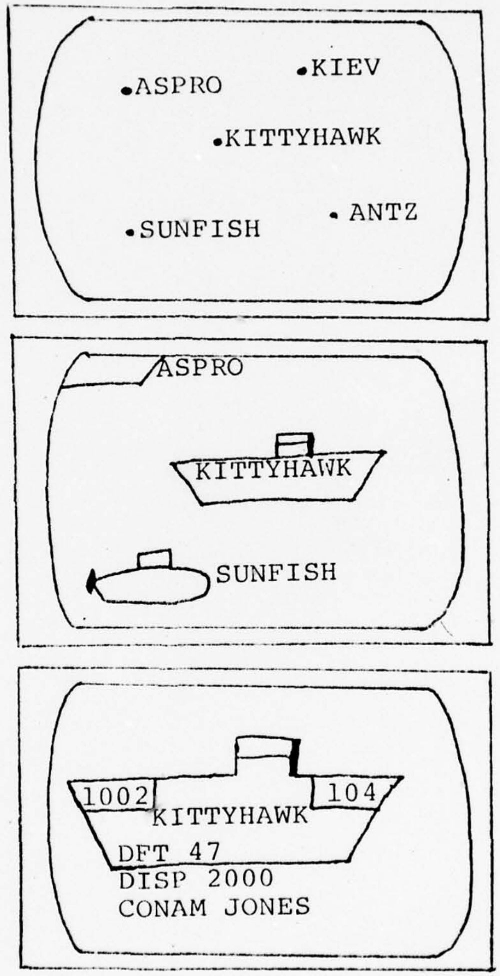
2.1 Zooming and Activation

While pressing of the zooming joystick at first causes the display to zoom, holding it down continuously soon causes something else to happen: a new image plane appears. This change of displayed information may be orchestrated to give the appearance of either of two situations: (1) more detail appears; or (2) the user enters a new information space, different than the one he left.

The incremental addition of detail can be carried on indefinitely, given sufficient capacity to store the images. By supplying new image planes, correctly aligned with their coarser predecessors, the illusion is maintained of having an infinite resolution photograph. Such a facility would allow great changes in apparent distance. For example, an observer could start at the sun and zoom in on a picture of earth until he was reading the fine print on a newspaper lying on the ground. More practically, this technique would be used to add symbolic information as the space became available to display it. For example, in a map of the Pacific ocean, a ship could be represented by a dot. As the user approached it, its

icon would change to the shape of a ship. As the observer continued to approach, text strings giving the name of the commanding officer and the ship's destination would appear.

Incremental Addition of Detail Figure 2.1



Alternatively, zooming in on an icon may cause the user to enter a totally new set of image planes, organized as

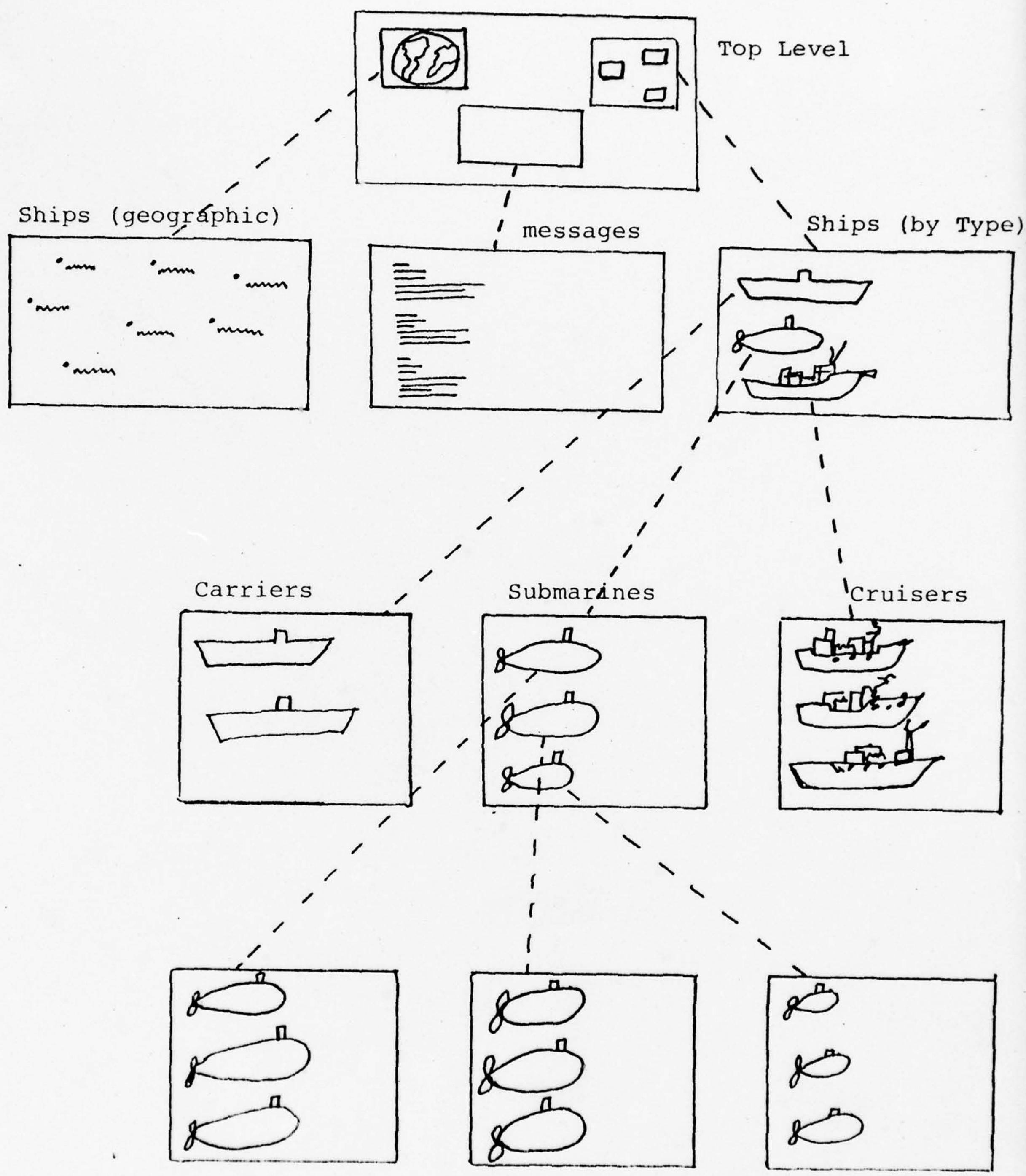
another information space (I-space). This information space could lead to yet more I-spaces, some of which, in turn, could be more sets of image planes, and some of which could, instead, cause the activation of various programs. Such programs could display animated sequences, allow the user to send and receive messages, or perform any other activity allowed by the computer system. In any case, the third monitor is available for the use by such programs. In the event that the I-space which is active is another plane of icons, this monitor would show the entire plane, as at the top level, or it could show a tree structure of where the user was in the nesting of I-spaces.

2.2 Icons and the Database Management System

Icons are created by various methods. They may be created by hand, as in the MIT system, through the use of a program for "painting" with the data tablet. However, if the user has a large number of similar data items, he may wish to use a standard icon, for instance a rectangle, and merely supply the position for each new instance. This process may be aided by a simple icon generation language which allows graphical properties of icons to be

Multiple Information Spaces

Figure 2.2



determined by properties in a symbolic database of the object indicated by the icon. This process is described in more detail in Section 5.

3. DISPLAY HARDWARE

The selection of a suitable display presented somewhat of challenge. The primary requirements for the display were:

1. 480 lines of 640 elements, for compatability with standard television
2. 8 bits of data per point, expandable to 16
3. zooming in integer scale factors (1,2,3,...16)
4. scrolling in single memory pixel steps
5. fast picture update times in vertical as well as horizontal update modes
6. 3 independendent display chanels on one controller
7. 24 bit color look-up table (8 bits each of red, green, and blue)

A specification of the required features was sent to six different manufacturers of raster scan color display hardware. While there were many interesting responses, no one company could provide all of the features required in an off-the-shelf product within the given time frame.

The vendor which was finally chosen, Lexidata Incorporated of Burlington, Massachusetts, was able to meet the

specification through some minor modifications of one of their standard products. This was possible because their product's architecture was well suited to the SDMS application, most notable through the provision of a "pixel packer" for rapid update of the display and the use of a microprogrammed display controller.

The pixel packer alleviated one of the most serious shortcomings of most of the displays surveyed: the time required to update the display memory. Most such displays achieve fast update times by favoring horizontal writes (a raster at a time) over vertical ones (a column at a time). This is done by interleaving the memory, such that one write operation typically modifies 8 or 16 picture elements (pixels). While, in theory, such interleaving should allow a column 8 or 16 pixels wide to be written at the same rate, many manufacturers did not have the provision for such operations. The Lexidata machine, with relatively trivial modification, was capable of doing this. Another feature of the Lexidata display made it suitable for SDMS: the microprogrammed display readout. The display uses a proprietary microprocessor to control which memory locations appear at which positions on the screens. Zooming and scrolling were thus simple to implement. The smooth horizontal scrolling was a bit more difficult, since the 16 pixel interleaving of the memory

allowed the location of the first pixel on the screen to be specified only to the nearest 16 locations. This was fixed by adding a new parameter to the program which moves the displayed position of the image in single pixel increments, combined with a feature which blanks out the remaining pixels. A complete specification of the display is included as Appendix A.

4. STAGING MANAGER

4.1 Overview

This document is intended to give an initial specification of the Staging Manager for display data for SDMS. It is intended to be expanded and revised as the design of the SM and the project is refined.

Information to be displayed must be moved from the disk to the 11's memory, possibly expanded or changed there, and then written to the display as it is to be seen. The organization of the information on the disk is physically linear and conceptually 2- or 3- dimensional, depending on our view of the universe. On the display, the organization is 2-dimensional, both physically and conceptually; and it is in a different coordinate framework from the universe. The mapping between these frameworks will be simple, but not constant, since the screen moves around in the universe.

Data must be presented to the display in rectangles, one byte per pixel. This implies that all data to be displayed must first be in main memory. Further, to avoid incurring disk delays every time the screen is moved in the universe, a margin of information must be maintained around that actually contained on the screen, which can be written to the display immediately when it is needed.

The SM is responsible for maintaining a copy of information to be displayed in memory in response to commands from the rest of SDMS, and presenting that information to the display as needed.

4.2 Tiles, Screen, Universe

We have considered a planar universe approximately 6500 pixels square; this is approximately one-half of an RP04 pack. The extension to a 3-dimensional universe is straightforward. The extension to a much larger universe (increasing the area by a factor of 8 or more) will probably affect the addressing scheme, but use of a high-order "universe-index" should suffice to carry through the ideas presented here.

To mediate between the physical linearity of disk storage and the conceptual 2-dimensionality of the universe, we partition the universe into Tiles. (In a 3-dimensional universe, a Brick can be used analogously.) Each tile contains all the display data for a small rectangle. The tiles will be ordered on the disk like the elements of an array, in horizontal stripes across the universe. It appears at present that the data for a tile will fit easily in a disk block; this is covenient, but not necessary. In memory, those tiles which cover the current screen plus an acceptable margin around the edges will be maintained in a byte-per-pixel format. Within a tile in memory, the bytes are again organized in array fashion, in horizontal stripes across the tile.

The optimum dimensions of a tile are debatable. Factors affecting the decision include: the amount of memory available for buffering the screen + its margin, the relative penalties for seeking vs. reading blocks on the disk, and the density with which display information is packed in a block. Assuming one byte per pixel on the disk and no more than 512K available for buffers, we have used a computer simulation to study the problem and have arrived at tile dimensions at or below 128 pixels on a side, with encouraging results. For instance, with tiles 80 X 106 pixels, it should take about 1.13 seconds to

stage the data to scan completely across the screen. There are tradeoffs which can be used to improve this performance considerably; for instance, restricting ourselves to scale 2 for high-speed scrolls reduces that time to less than a quarter: at scale 2, for blocks 128 X 96, a complete horizontal scroll should take about .27 seconds.

The density factor may be expected to be much less for pictures which are generated from some simpler description (e.g. run-length encoding). Nonetheless, tile sizes will probably be bounded fairly strictly by the core buffer size. Considering the uncertainty surrounding tile dimensions, it seems well to isolate tiles as much as possible from the rest of the system.

4.3 Buffers

An image of the data on the screen will be maintained in the 11's memory, along with an image of information which is off, but near, each edge of the screen. This is the Virtual Frame Buffer (VFB). It will be organized by tiles. Given the tile dimensions under consideration, it will be about 6 tiles square, and require close to 512K bytes. Associated with the VFB is a table which associates a tile-id with the corresponding area in physical core (the VFBMAP). The SM will be responsible for manipulating the 11's memory management to access all of the data in the VFB. In general only one tile will be manipulated at a time; even so, it will probably require several page address registers (PARs) of the PDP11/70's memory mapping unit to address the whole of a tile.

The organization of the VFBMAP is an ordered table of pairs

tile-id : page-no

where tile-id is a number identifying the tile in the universe, and page-no is the high-order 9 bits of the

physical address of the tile's origin (this is the quantity which gets loaded into the appropriate PAR). The values of the page lengths stored in the page descriptor registers (PDRs) are constant across tiles. Any given tile may require more than one PDR value, but these will correspond for all tiles. PARs to address other than the first page of an individual tile can be figured by the addition of a constant.

The VFBMAP will probably be maintained in order; that is, its contents will be shuffled when a new stripe is staged horizontally or vertically. This is not really an important point, but it looks like it will ease conversions between screen and tile coordinates.

Separate buffer areas will be needed to provide data to the display in the format it needs. This format consists of rectangles whose horizontal dimension is a multiple of 16; the pixels of the rectangle placed in contiguous horizontal stripes. Since this format will often agree with neither the tile division of the universe, nor the boundaries of the objects depicted, the strategy is to construct the picture in the VFB, and then copy the relevant portions, one tile at a time, into separate display buffers, which are then fed to the display.

In the case where information on the disk does not correspond to simple images, provision will also have to be made for disk buffers separate from the VFB, so that routines in SDMS may read the concise definition of the picture area and expand it into the pixels in the VFB. The intention is that other parts of SDMS will manipulate the view on the screen by making calls to SM which manipulate the VFB. These calls will be able to specify the relevant areas by either their screen or universal coordinates. As a side effect, these calls will provoke commands to the display, and often disk accesses as well.

4.4 Display Parameters

SM must maintain the following information about the current state of the display:

- its origin: X and Y universe coordinates of the screen's origin;

- its dimensions: the actual currently displayed length and width of the screen;

- velocity in X and Y may be needed for proper scheduling of staging.

Information about the tile location of the screen origin will prove extremely useful:

- the tile in which the origin is currently located; normally this will be the 0th resident tile, but the origin can move a margin-width outside this tile in each dimension. It probably will be kept as the index in the VFBMAP of the origin's tile.

- tile coordinates of the origin: the X- and Y-offsets into the tile's data of the screen origin.

- tile address of the origin: the byte address within its tile of the screen's origin.

4.5 Conversions

This section gives the most common conversions among different ways of looking at the same data; presumably these will be incarnated in appropriate procedures or macros at the appropriate time. Division truncates; <u>rem</u> is the remainder operator; a number of constants are predefined:

tile address:	the virtual address of the first pixel in a given tile
tile_len:	horizontal tile dimension in bytes
tile_ht:	vertical " " " "
urow:	the number of tiles in a horizontal stripe across the universe
xorg:	the universal coordinates of the
yorg:	screen origin

```
tile_id(x,y)
/* x and y are the universal coordinates of a point
   this routine returns the id of the tile which
   contains that point.
```

```
 */
        ( (y / tile_ht) * urow ) + (x / tile_len) ;


tile_org(t)
/* given a tile-id, returns the universal
   coordinates of that tile's origin as a
   pair [s, x,y]
 */
        [ (t rem urow) * tile_len ,
          (t / urow) * tile_ht      ]


tile_addr(x,y)
/* given the universal coordinates of a point,
  return its physical address
 */
        int tx, ty, t
        t := tile_id(x,y)
        [ tx, ty ] := tile_org ( t )
        tile_address  + tile_org(t) + ( (y-ty)*tile_ht ) + x - tx ;
```

To translate a pair of screen coordinates to universal coordinates, add xorg and yorg; in the opposite direction, subtract.

4.6 Calls

Any call to SM is likely to change the picture on the screen; some may also cause staging.

When an object which is already displayed is to be changed, no additional data will be displayed, but SM should still be informed, so that the VFB is updated appropriately. This is necessary because that object may be scrolled outside the screen and later scrolled back on. In the meantime, the new value will have to be preserved; perhaps written to disk, or perhaps merely retained in core, but in any case, SM must feed the proper (updated) information to the display. A possible exception to this principal is a change to the color table, which probably doesn't affect SM at all. Routines which change the display without the possibility of causing staging include things like DRAW and FILL, which put new values in existing areas of the display. These will define a rectangle, probably in screen coordinates, and a source for the new value. Also in this category is a resetting of the screen's dimensions, because it may affect how much data SM feels required to feed to the display.

The operation which can trigger the display is some motion of the screen. This may be done either absolutely (SET_ORG(x,y)), or relatively (SCROLL (dx, dy)). In response to such a call, SM must

1. determine if the move requires staging; if so this will be done for one stripe, and then, if necessary, for the other:

   a. it shuffles the VFBMAP to correspond to the new screen origin. This has a side effect of putting the physical addresses of the to-be-filled buffers (the free space in memory) in the slots of the tiles which will eventually use them.

   b. computes the tile-ids of the tiles in the new stripe, and inserts them in the proper slots in the VFBMAP

   c. reads each of the needed tiles from the disk, passing them to an expanding routine if necessary.

2. SM determines if new data is needed by the display; if so, it proceeds one tile at a time to build display buffer and pass it to the display.

3. this process is repeated, if necessary, for staging in the orthogonal direction.

This description has been written as though SM could be called in-line from some other part of SDMS, and eventually return when the display is again consistent. If this constraint is too strong, we must define what interlocks are needed between SM and the rest of SM.

5. INTERFACE TO DATABASE MANAGEMENT SYSTEM

This chapter discusses the key concepts of a proposed SDMS data model. It emphasizes the relationship between a symbolic database of a traditional type and a graphic presentation of that database.

5.1 Objectives

There are two specific objectives which this data model attempts to achieve:

1. To provide a mechanism for the automatic generation of graphic objects as the result of insertions (or updates) of new data in the symbolic database.

   It is clear that in many applications the SDMS user will not be the source for much of the data in his database. In command and control applications, for example, there are large databases, standardized across DOD with well-defined update channels originating in automatic sensors, human observers and unit commander reports. These result in

changes to a symbolic database. The rate of these changes is far too great for the SDMS user to be expected to decide in each case what the impact on the graphic representation should be. Instead there must be an automatic procedure, established when the database is defined, which generates the graphic change based on the nature of the update and the previous state of the database. The extent of the end user's interaction in this process is chosen according to the demands of the application and can range from total control: the selection of icon position, size, orientation and color; to no control at all: completely automatic definition of the graphic image.

2. To provide a mapping between the symbolic database and its graphic representation so that data objects selected symbolically can have their selection represented graphically (and vice-versa). Many classical database operations involve the selection of subsets of the database. Very often these operations are best expressed symbolically (or at least no good graphic technique for expressing them has been proposed), but the results of the operation are conveniently expressed graphically.

For example, consider a database of ships. Each separate ship is represented in SDMS by a graphic object visible in some region of space. The layout might be geographical, with position in the graphic space isomorphic to position in the ocean, or logistical, with position in the graphic space based on membership in a capability class (e.g., nuclear, non-nuclear). Suppose we wanted to know which ships are bound for the Mediterranean. We could define this graphically by zooming in on each ship to see its destination but this would be a very laborious process. It would be far simpler to state the problem symbolically, for example, by typing:

Blink ship where ship.destination = "Mediterranean"

The response to this query makes effective use of the system's graphic capabilities and, one suspects, is far more expressive to the user than a symbolic response such as a list of the corresponding ship identifiers. In this way, we achieve a very fruitful marriage of the symbolic and graphic capabilities of the system.

In order to accomplish this it is necessary to map between objects selected symbolically and their graphic representations. The data model defined here is intended to provide this mapping capability.

5.2 Symbolic database

The symbolic databases used in SDMS will be relational in format. This data model was chosen largely for convenience: a key target database for application of SDMS, the ACCAT database, is relational and the only suitable DBMS available on the PDP-11 is relational. In fact, even if these compelling practical considerations were not present, the relational model would have been selected because its simplicity and clarity make it the best suited of the existing data models for extension to the SDMS concept.

5.3 Graphic data space

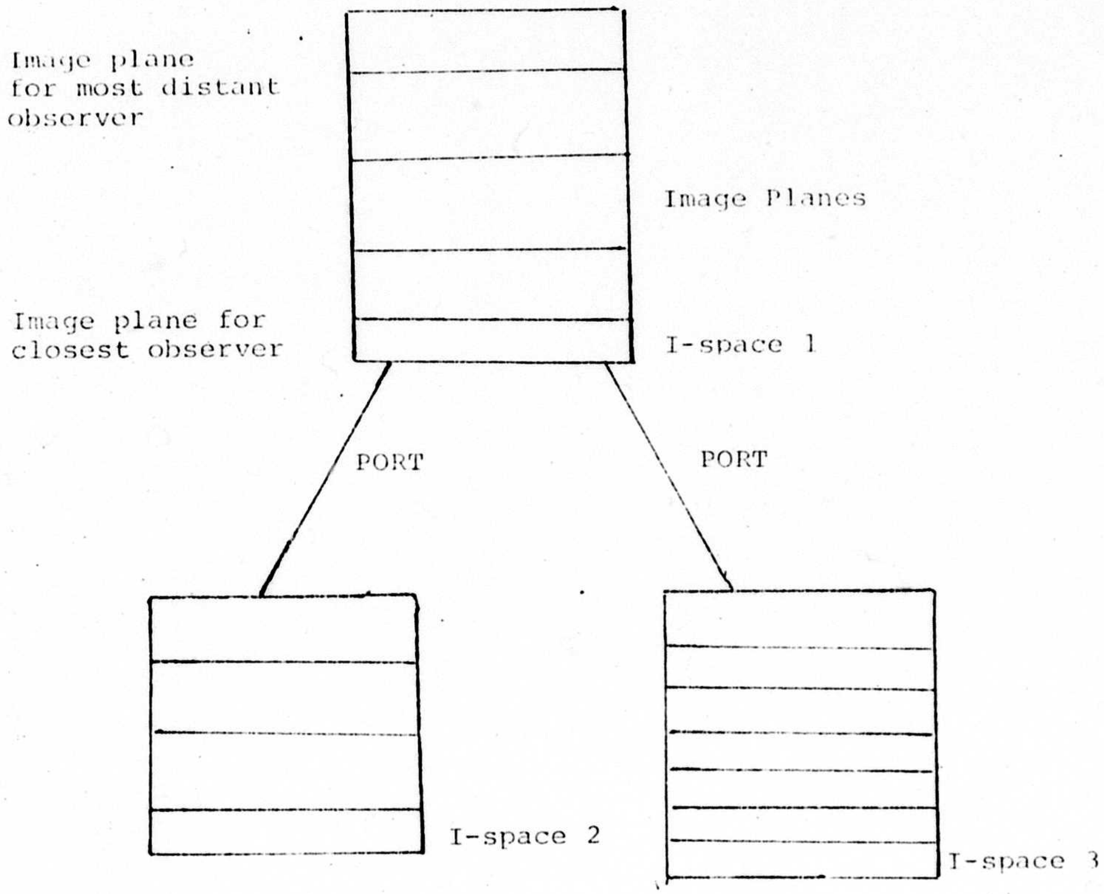
The graphic representation of the database is called the graphic data space or GDS. The GDS is a collection of independent 2-1/2 D information spaces called I-spaces. There are entry points called ports for travelling from one I-space to another. Each I-space contains graphic objects which can be seen at various levels of resolution depending on how closely the user has "zoomed-in" on the object. The amount of detail visible is defined in terms of image planes. Each I-space is composed of an ordered set of image planes representing the image visible to a viewer at different perceived magnifications of the viewing plane. For example, an I-space containing data about ships might consist of three image planes. The first plane, containing the image presented to the most distant observer, may represent each ship as a dot. The second plane, containing an image for closer observers, might show the basic shape of the ship in a silhouette. The closest observer sees the third plane and maximum detail. This detail might include text data giving the name of the Captain, the ship destination, its class, and

so forth. Figure 5.1 illustrates this organization of graphic data space.

5.4 Entities

The term entity will be used to refer to a tuple of the symbolic database whose insertion implies the creation of a graphic object in graphic data space and whose deletion implies the elimination of that graphic object. For example, in a database of ships, the tuples corresponding to individual ships (whose keys might be hull numbers) are entities. Many tuples are not entities; these tuples normally provide additional information about entities. For example, some tuples in the database about ships may provide descriptive information about standard ship classes. Each ship tuple will provide the names of the class of which the ship is a member so that the additional class description data can be found by referring to the appropriate class tuple. For organizational simplicity, we will require that either all tuples of a relation are entities or that none are. Thus, we use the term entity relation to refer to relations containing entities.

Graphic Data Space Figure 5.1



5.5 Important times

There are three distinct points in time which are relevant to displaying graphic images in SDMS:

a. the time that an entity relation is defined (definition time);

b. the time that an entity is inserted into the database (insertion time); and

c. the time a user views a portion of the graphic data space (retrieval time).

At definition time the standardized aspects of that class of entities which form a relation are established. For example, the rules for choosing the shape of ships by looking in a table indexed by CLASS would be established at definition time. No images are actually created at definition time.

At insertion time, when for example the ship with hull number 509 is placed in the database, the graphic image corresponding to the inserted entity is created. This is a "logical creation" implying that a user who views the portion of the graphic data space in which the graphic

image has been created will see that graphic image. It does not necessarily imply that a bit map representation of the image is stored. At retrieval time, the graphic image is constructed in bit map form and displayed.

5.6 Links

In addition to the mapping from symbolic data to graphic data represented by the entity-graphic image correspondence, there are mappings represented by constructs called links. A link shows a correspondence between a tuple in the symbolic database and an object in the graphic data space. This link is used to "excite" the graphic image when the tuple is selected in a symbolic search. This excitement is represented in some graphic form such as blinking or changing color. The link is also used to select a tuple when its corresponding graphic object is selected graphically, for example by touching or framing. There is always a link defined between an entity and its graphic image. Other links may be defined as part of the definition of the graphic image for entities of a given relation. For example, in defining the graphic image for ship a link might be established to the tuple corresponding to the class of the ship. In this way, if

we select all Kittyhawk class ships, there is a defined correspondence to graphic images, so that all of the instances of these ships can be made to blink.

5.7 Graphic definitions

Each relation whose tuples are entities has a graphic definition which defines (at definition time) the standardized aspects of those entities in the same relation. The definition is used to create a graphic image group for each tuple (i.e. entity) inserted into the relation. The group has one image for each plane of the I-space in which the entity exists. A graphic definition consists of a "graphic image definition" for each of these individual images. There will be one graphic definition corresponding to each I-space in which the entity appears.

A graphic image definition has 5 parts corresponding to 5 aspects of the image to be created: icon, position, orientation, size, color. The term icon refers to the graphical symbol itself independent of its position, orientation, size, and color. The definition of the icon is the most complicated part of the graphical definition. The following description of how this graphic aspect is handled will be suggestive of the scheme for handling the other 4 aspects as well.

An icon definition is a function represented as ICON\R^(KEY) where R is the name of the entity relation and KEY is its primary key. The intent of the KEY argument is to provide an entry point into the symbolic database to select values used in defining the icon. For example suppose there is a table of flags from which one will be selected to place on a given ship icon by using the function FLAG(NATIONALITY). The NATIONALITY to be used in this expression can be found by following KEY to the appropriate attribute.

The icon definition is a program written in a graphics language. This language will include the following features (in addition to the usual graphics operations):

a. EQUEL statements (an interface to INGRES for extracting information from the database to use in selecting graphic symbols);
b. LINK statements (for defining links);
c. references to a library of graphic subroutines (e.g. SHIP-SHAPE(CLASS))
d. LIKE statements (for defaulting some aspect of the icon to that shown on a different image plane).

The graphic definition may indicate that some element of an image will be left for definition by the end user. For example, position may be treated in this way. This results in the image being created in a default location

and making it available to the user for movement to a location of his choice.

5.8 Defaults

Since the creation of elaborate graphic definitions may be a laborious process, it is important that good defaults be chosen so that simple graphic definitions can be created simply. It will, of course, be easy to experiment with a variety of defaults when the system is in operation. We should plan to do this experimentation so that real confidence in the defaults can be gained. The following list of preliminary defaults is presented as a plausibility argument that effective defaults can be created.

Defaults for graphic image definition:

icon - rectangle containing key of entity

position - set by end user

orientation - parallel to boundaries of I-space (and screen)

size - chosen for comfortable reading of key by most distant observer

color - set by end user from palette

A.

FRAME BUFFER SPECIFICATION

This is a specification for a framestore raster scan display for use with a Digital Equipment Corporation PDP-11/70 minicomputer. It will store, to a precision of eight bits per point, at least one full frame of digital, computer generated data that will ultimately provide a full color video signal. It must meet the following specifications:

Video Output: The system shall provide Red, Green, and Blue analogue video outputs with amplitude 1 volt P-P into 75 ohms, and capable of driving a standard video color monitor (Barco model CDCT2/51-H or equivalent). Synchronization signals shall be provided on a separate output, 1 volt minimum P-P. All video and sync connections shall be made with BNC connectors.

Synchronization shall be RS-170 compatible, full interlaced scan, and the system shall have the provision to be synchronized to a standard set of input signals as

provided by a NTSC sync generator (Tektronix model 1470, or equivalent).

The output format shall be a video signal of 480 visible lines of 640 picture elements per line.

Independent Display Channels: The system shall provide three independent color output channels, referred to here as Channels 1, 2, and 3. Channel 1 shall contain 8 bits of memory per pixel, expandable in the future to 16 bits. Channels 2 and 3 shall each contain 4 bits of memory per pixel, expandable to 8 bits each. It shall be possible under software control to reconfigure the system so that channels 2 and 3 display the same picture from an 8 bit per point image.

Storage Capabilities: Sufficient memory shall be provided for each channel to contain a picture that has 487 lines of 672 pixels.

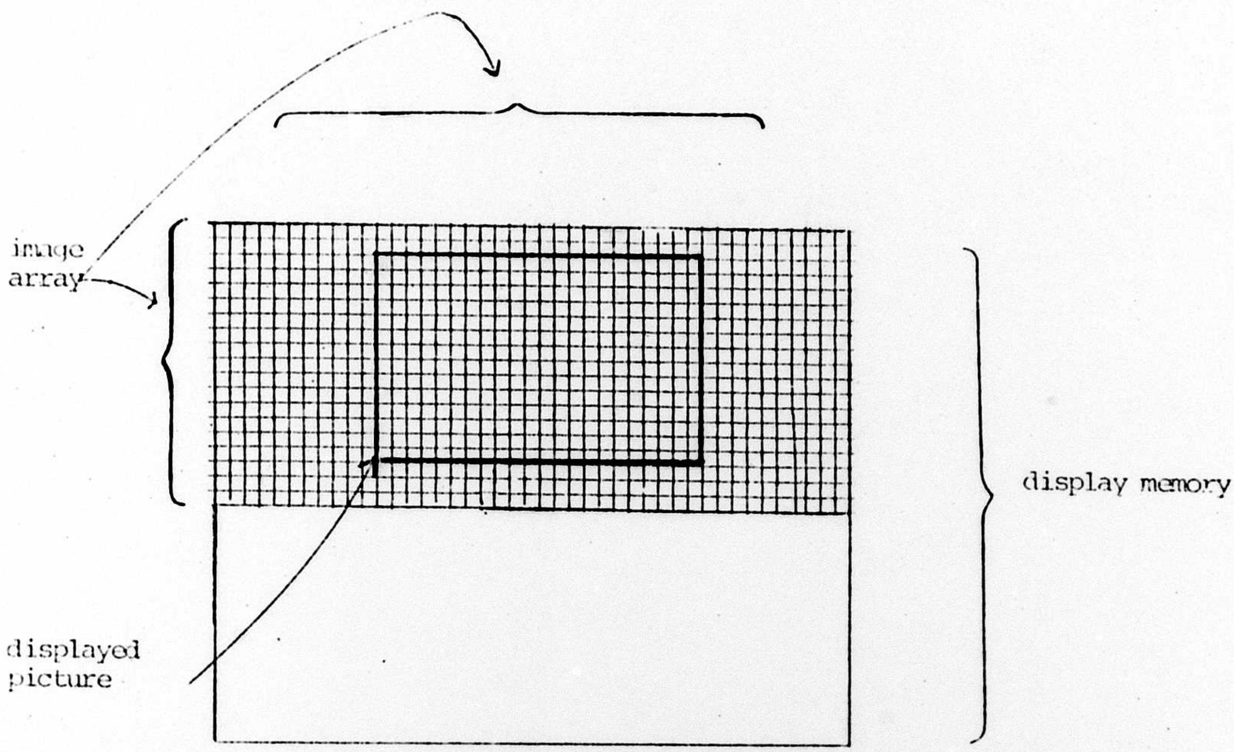
The system shall have the ability to display a "zoomed in" version of the stored image, with no change to the internal data stored in the frame buffer. This shall be accomplished by repeating picture elements and lines as required, with the scale factors controllable to integer precision from one (full scale) to 16.

The system shall allow the display memory to be formatted into a two dimensional image array having a width variable from 16 to 672 pixels and a height variable from 1 to 487 lines, subject to the size of the memory. Pixels shall be stored contiguously in memory. It shall be possible to display any portion of the image array by specifying the start point in display memory, the number of lines, the number of pixels per line, and the location in screen pixel coordinates of the first pixel to be displayed. Note that while the image array is contiguously stored, the displayed picture might not be, resulting in parts of the image array not being displayed. This condition shall in no way effect the integrity of the stored image (i.e. non-displayed dynamic memory must still be refreshed).

Refer to Figure A.1: It shall be possible to set the location in display memory of the image array and the displayed image. Note that moving location by a line length results in vertical scrolling, while moving them by lesser amounts results in horizontal scrolling. It shall be possible to perform such motion in either of two modes:

1. Display memory wraps around from line to line and from bottom to top. (Figure A.2).

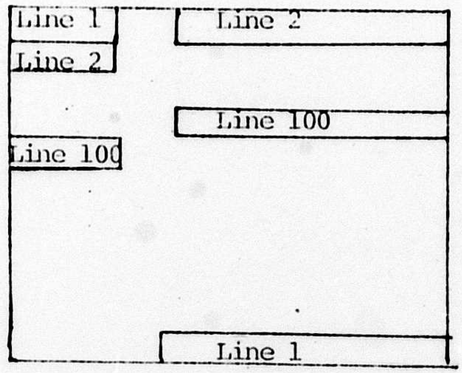
Figure A.1



2. No wrap-around, with portions of the screen fallng outside of the image array being displayed as black. (Figure 3.)

A mechanism shall be provided for moving the image on a memory pixel basis, to give the appearance of a smooth scroll with no change in the size and location of the active screen area and its borders.

Figure A.2

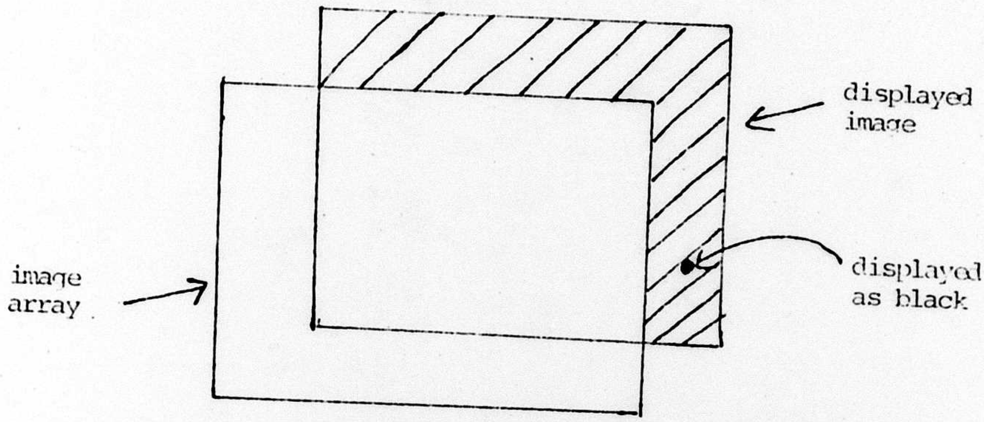


All modification to the readout such as zooming, scrolling, and aspect ratio modification shall be made so that at no time is there visible image break-up on the screen, and no loss of synchronization. All modifications to the readout shall be achievable in one frame time.

Zooming and scrollng channel 1 shall have no effect upon channels 2 and 3, and vice-versa. It is permissable for channels 2 and 3 to scroll and zoom together.

Color Capabilities: The system shall provide a look-up table for each of the three channels which translates the

Figure A.3



value of each pixel stored in the image array into at least eight bits each of Red, Green and Blue. This table shall be readable and writeable from the host computer, and any modification to it shall result in no visible artifact on the screen. The entire table shall be writeable in 50 milliseconds.

Host PDP-11/70 Interface: The display system shall be capable of being interfaced to a Digital Equipment PDP-11/70 via a DR70 parallel interface, or equivalent. The host shall have the capability to read or write the

image stored in the array in a random access fashion, accessing any pixel in less than 60 microseconds. Additionally, it must be able to update pixels which lie in a rectangular window at a rate no worse than one microsecond per pixel. This speed may be achieved by requiring that vertical window edges lie on sixteen pixel boundaries, and have widths that are multiples of sixteen pixels.

The system shall calculate display memory addresses from coordinate information provided by the PDP-11. All software which is necessary to perform the above function, and which will run in the display controller shall be provided.

The system shall have an "overlay" plane, consisting of a single bit per point image which may be combined with the above described channels by means of color look-up table.

The following documentation shall be provided:

1. A User's Manual which describes the operation of the system and the programming interface.
2. A maintenance manual describing how the hardware works and how it can be adjusted.

3. Complete circuit diagrams.
4. An installation guide describing all relevant details of unpacking and mounting the unit and connections to A/C power, video devices, and the host computer.

References

[FIELDS and NICHOLAS]
Fields, Craig; and Negroponte, Nicholas, "Using New Clues to Find Data", Third International Conference of Very Large Databases, Tokyo, Japan, 1977.

[BOLT]
Bolt, Richard A., Spatial Data-Management, Interin Report, MIT Architecture Machine Group, November, 1977.

[DONELSON]
Donelson, William C., "Spatial Management of Multimedia Information", to appear in proceedings of SIGGRAPG '78, August 1978.